

1 **XPATH EVALUATION AND INFORMATION PROCESSING**

2 **FIELD OF THE INVENTION**

3 The present invention relates to a technique for
4 analyzing an Extensible Markup Language (XML) document. More
5 specifically, the present invention relates to an analyzing
6 system for evaluating the XML Path Language (XPath) and an
7 analyzing method thereof.

8 **BACKGROUND OF THE INVENTION**

9 The XML which can freely define a logical structure of
10 a document and significance of a constituent is widespread
11 today as a document description language for use in packet
12 switching on a computer network represented by the Internet.
13 When an XML document is used in an application program, it is
14 necessary to analyze the XML document to be processed. The
15 XPath is used in this event. The XPath is a language for
16 defining a sentence which indicates a specific part of an XML
17 document, which is the specification recommended by the World
18 Wide Web Consortium (W3C). By use of the XPath, it is

1 possible to analyze whether or not a given XML document
2 includes a specific logical structure or a constituent (see
3 Non-patent Literature 1, for example).

4 A structure of an XML document can be expressed by a
5 tree structure. An XML document is expressed by use of the
6 Document Object Model (DOM), which is the specification
7 recommended by the W3C, or the like. When the XML document
8 is analyzed by use of the XPath, the entire XML document
9 expressed in the DOM or the like is usually subjected to
10 scanning to check whether the XML document has a structure
11 described with the XPath or not. Therefore, an analyzing
12 system for XML documents needs to read the entire XML
13 document to be analyzed into a given memory in order to
14 perform processing.

15 As described above, when an XML document is analyzed by
16 use of the XPath, for example XML Path Language (XPath)
17 Version 1.0, it has been conventionally necessary to read the
18 entire XML document to be analyzed once into a work area of a
19 memory. However, when the data size of the XML document to
20 be analyzed is huge, large memory usage is required to read
21 the XML document as description of the DOM or the like.
22 Moreover, a lot of time is required for processing of genera-
23 tion and operation of the DOM.

24 In the case of using a streaming-based application

1 program interface (API) such as the Simple API for XML (SAX)
2 or the Xerces Native Interface (XNI), it is inefficient that
3 the processing can be started only after reading the entire
4 XML document once in order to perform an analysis by use of
5 the XPath, in spite of the fact that the API can perform
6 serial processing of the XML document obtained in accordance
7 with a streaming format.

8 **SUMMARY OF THE INVENTION**

9 In consideration of the foregoing problems, it is an
10 aspect of the present invention to realize systems, apparatus
11 and analyzing methods for evaluating the XPath while subject-
12 ing an XML document to streaming processing. To attain the
13 foregoing aspect, the present invention is realized as an
14 XPath evaluating method for evaluating the XPath relevant to
15 an XML document by use of a computer. In an example embodi-
16 ment, this XPath evaluating method comprises a first step of
17 serially inputting XML event strings constituting an XML
18 document to be processed, a second step of serially evaluat-
19 ing the XPath relevant to the respectively inputted XML
20 events and retaining information concerning a result of
21 partial evaluation of the XPath in given storing means when

1 the XPath is partially established with respect to a given
2 XML event, and a third step of repeating the partial evalua-
3 tion of the XPath along with the input of the XML event
4 strings while considering the result of the partial evalua-
5 tion retained in the storing means and judging that the XPath
6 is established with respect to the XML document to be
7 processed when the last part of the XPath is established.

8 Another aspect of the present invention is realized as
9 an XPath evaluating apparatus having the following configura-
10 tion. This XPath evaluating apparatus includes an evaluation
11 executing unit for executing evaluation of the XPath by
12 streaming processing, and an XML event transferring unit for
13 inputting XML event strings which constitute an XML document
14 to be processed and serially transferring the XML event
15 strings to the evaluation executing unit. This evaluation
16 executing unit serially evaluates the XPath with respect to
17 each of the XML events transferred from the XML event trans-
18 ferring unit and retains information concerning a result of
19 partial evaluation of this XPath with respect to a given XML
20 event when this XPath is partially established. Moreover,
21 the XPath evaluating apparatus judges that the XPath is
22 established in this XML document when the last step of this
23 XPath is established.

24 Another XPath evaluating apparatus according to the

1 present invention includes a document tree constructing unit
2 for inputting XML event strings which constitute an XML
3 document and serially constructing a document tree indicating
4 a document structure of the XML document based on inputted
5 XML events along with the input of the respective XML events,
6 an XML event transferring unit for inputting the XML event
7 strings which constitute the XML document to be processed and
8 serially transferring the XML event strings to the document
9 tree constructing unit, and an evaluation executing unit for
10 evaluating the XPath along with construction of the document
11 tree by this document tree constructing unit while using the
12 document tree with a part which has been constructed.

13 Furthermore, the present invention can be also realized
14 as an information processing apparatus including an XML
15 parser, an XPath evaluating unit having a function of the
16 above-described XPath evaluating apparatus, and an applica-
17 tion executing unit for inputting an XML event generated by
18 the XML parser and executing processing of an XML document
19 configured by the inputted XML events in response to an
20 evaluation result of the XPath by the XPath evaluating unit.

21 **BRIEF DESCRIPTION OF THE DRAWINGS**

1 For a more complete understanding of the present inven-
2 tion and the advantages thereof, reference is now made to the
3 following description taken in conjunction with the accompa-
4 nying drawings, in which:

5 Fig. 1 is a view schematically showing an example of a
6 hardware configuration of an information processing apparatus
7 which is suitable for realizing Embodiment 1 of the present
8 invention;

9 Fig. 2 shows an example of loading XPath evaluating
10 means (an analyzing system) in Embodiment 1;

11 Fig. 3 is a view describing a functional configuration
12 of an XPath evaluating unit in Embodiment 1;

13 Fig. 4 is a view schematically showing an automaton
14 generated by an automaton generating unit in Embodiment 1;

15 Figs. 5A to 5C are views showing examples of a location
16 path, an automaton, and an XML document to be processed which
17 are used in processing according to Embodiment 1;

18 Fig. 6 is a flowchart describing streaming processing
19 of the XML document according to Embodiment 1;

20 Fig. 7 shows an example of loading XPath evaluating
21 means (an analyzing system) in Embodiment 2;

22 Fig. 8 is a view describing a functional configuration
23 of an XPath evaluating unit in Embodiment 2;

1 Fig. 9 is a view exemplifying relations between types
2 of axes and stacks generated in this embodiment;

3 Fig. 10 shows stacks generated by the XPath based on
4 the relations shown in Fig. 9;

5 Fig. 11 shows an example of an XML document to be
6 processed;

7 Figs. 12A to 12C are views showing aspects of evalua-
8 tion processing by use of stacks for the XML document of Fig.
9 11;

10 Figs. 13A to 13C are views showing aspects of evalua-
11 tion processing by use of stacks for the XML document of Fig.
12 11;

13 Figs. 14A and 14B are views showing aspects of evalua-
14 tion processing by use of stacks for the XML document of Fig.
15 11;

16 Fig. 15 shows another example of an XML document to be
17 processed;

18 Fig. 16 shows the XPath to be evaluated and a stack
19 generated from this XPath;

20 Figs. 17A to 17C are views showing aspects of evalua-
21 tion processing by use of stacks for the XML document of Fig.
22 15;

23 Figs. 18A to 18C are views showing aspects of evalua-
24 tion processing by use of stacks for the XML document of Fig.

1 15;

2 Figs. 19A and 19B are views showing aspects of evalua-
3 tion processing by use of stacks for the XML document of Fig.
4 15;

5 Fig. 20 is a flowchart describing streaming processing
6 of the XML document according to Embodiment 2;

7 Fig. 21 shows an example of loading XPath evaluating
8 means (an analyzing system) in Embodiment 3;

9 Fig. 22 is a view describing a functional configuration
10 of an XPath evaluating unit in Embodiment 3;

11 Fig. 23 is a flowchart showing procedures for evaluat-
12 ing the XPath by an evaluation executing unit of Embodiment 3
13 using a document tree and a saved set, which shows an opera-
14 tion when a start tag token is inputted as an XML event;

15 Fig. 24 is a flowchart showing the procedures for
16 evaluating the XPath by the evaluation executing unit of
17 Embodiment 3 using the document tree and the saved set, which
18 shows the operation when the start tag token is inputted as
19 the XML event;

20 Fig. 25 is a flowchart showing procedures for evaluat-
21 ing the XPath by the evaluation executing unit of Embodiment
22 3 using the document tree and the saved set, which shows an
23 operation when an end tag token is inputted as the XML event;

24 Fig. 26 shows an XML document to be processed, SAX

1 event strings thereof, and the document tree constructed in
2 this embodiment;

3 Fig. 27 shows an aspect of a SAX event string, a
4 document tree, and transition of a saved set with respect to
5 an XML document, which shows the aspect when a SAX event
6 "startDocument" is inputted;

7 Fig. 28 shows an aspect of SAX event strings, the
8 document tree, and transition of the saved set with respect
9 to the XML document, which shows the aspect when a SAX event
10 "startElement:"a" is inputted;

11 Fig. 29 shows an aspect of SAX event strings, the
12 document tree, and transition of the saved set with respect
13 to the XML document, which shows the aspect when a SAX event
14 "startElement:"c" is inputted;

15 Fig. 30 shows an aspect of SAX event strings, the
16 document tree, and transition of the saved set with respect
17 to the XML document, which shows the aspect when a SAX event
18 "endElement:"c" is inputted;

19 Fig. 31 shows an aspect of SAX event strings, the
20 document tree, and transition of the saved set with respect
21 to the XML document, which shows the aspect when a SAX event
22 "startElement:"b" is inputted;

23 Fig. 32 shows an aspect of SAX event strings, the
24 document tree, and transition of the saved set with respect

1 to the XML document, which shows the aspect when a SAX event
2 "endElement:"b"" is inputted;

3 Fig. 33 shows an aspect of SAX event strings, the
4 document tree, and transition of the saved set with respect
5 to the XML document, which shows the aspect when a SAX event
6 "startElement:"d"" is inputted;

7 Fig. 34 shows an aspect of SAX event strings, the
8 document tree, and transition of the saved set with respect
9 to the XML document, which shows the aspect when a SAX event
10 "endElement:"d"" is inputted;

11 Fig. 35 shows an aspect of SAX event strings, the
12 document tree, and transition of the saved set with respect
13 to the XML document, which shows the aspect when a SAX event
14 "endElement:"a"" is inputted;

15 Fig. 36 shows an aspect of SAX event strings, the
16 document tree, and transition of the saved set with respect
17 to the XML document, which shows the aspect when a SAX event
18 "endDocument" is inputted; and

19 Fig. 37 is a flowchart describing streaming processing
20 of the XML document according to Embodiment 3.

21 **DESCRIPTION OF THE INVENTION**

1 The present invention provides methods, systems and
2 apparatus for evaluating the XPath while subjecting an XML
3 document to streaming processing. In an example embodiment,
4 this is realized as the following XPath evaluating method for
5 evaluating the XPath relevant to an XML document by use of a
6 computer. This XPath evaluating method comprises a first
7 step of serially inputting XML event strings constituting an
8 XML document to be processed, a second step of serially
9 evaluating the XPath relevant to the respectively inputted
10 XML events and retaining information concerning a result of
11 partial evaluation of the XPath in given storing means when
12 the XPath is partially established with respect to a given
13 XML event, and a third step of repeating the partial evalua-
14 tion of the XPath along with the input of the XML event
15 strings while considering the result of the partial evalua-
16 tion retained in the storing means and judging that the XPath
17 is established with respect to the XML document to be
18 processed when the last part of the XPath is established.

19 In this way, the evaluation of the XPath is carried out
20 individually concerning inputted XML events, and judgment of
21 establishment of the entire XPath is made while accumulating
22 the result of partial evaluation. Accordingly, the evalua-
23 tion of the XPath by use of streaming processing becomes
24 possible.

1 As concrete modes for retaining the evaluation and the
2 result of partial evaluation of the XPath performed in the
3 second step, the present invention discloses a mode using an
4 automaton, a mode using stacks, and a mode using a document
5 tree which is serially constructed along with the input of
6 the XML event strings.

7 In the mode using an automaton, an automaton for
8 expressing the XPath to be evaluated is generated, and the
9 XPath is evaluated by allowing transition of a state of this
10 automaton based on the respective XML events. The result of
11 this partial evaluation is retained as the state of the
12 automaton.

13 In the mode using stacks, a first stack which expresses
14 the XPath is generated to be evaluated with a string of stack
15 elements. Meanwhile, a second stack for analyzing a nested
16 structure of the XML document to be processed based on the
17 respective XML events is generated. Then, the XPath is
18 evaluated by comparing the first stack with the second stack.
19 The result of this partial evaluation is retained as the
20 stack elements of the second stack.

21 In the mode using the document tree serially
22 constructed along with the input of the XML event strings,
23 the document tree indicating a document structure of the XML
24 document to be processed is serially constructed based on the

1 input of the respective XML events. Along with construction
2 of this document tree, the XPath is evaluated by use of the
3 document tree with a part which has been constructed. In
4 this case, since the document tree is constructed along with
5 the input of the XML event strings, the XPath is evaluated
6 every time when a new node is added to the document tree so
7 as to check whether the XPath is established or not.
8 However, there is also a case where a plurality of nodes
9 satisfying establishment of the XPath exist. Accordingly,
10 the information concerning the result of partial evaluation
11 is retained so that the information can be used to evaluate a
12 subsequent XML event.

13 The present invention also provides methods, systems
14 and apparatus to realize an XPath evaluating apparatus having
15 the following configuration. This XPath evaluating apparatus
16 includes an evaluation executing unit for executing evalua-
17 tion of the XPath by streaming processing, and an XML event
18 transferring unit for inputting XML event strings which
19 constitute an XML document to be processed and serially
20 transferring the XML event strings to the evaluation execut-
21 ing unit. This evaluation executing unit serially evaluates
22 the XPath with respect to each of the XML events transferred
23 from the XML event transferring unit and retains information
24 concerning a result of partial evaluation of this XPath with

1 respect to a given XML event when this XPath is partially
2 established. Moreover, the XPath evaluating apparatus judges
3 that the XPath is established in this XML document when the
4 last step of this XPath is established.

5 In order to realize a mode using an automaton as a
6 concrete mode for retaining the evaluation and the result of
7 the partial evaluation of the XPath described above, this
8 XPath evaluating apparatus may further include an automaton
9 generating unit for generating an automaton which expresses
10 the XPath. Similarly, in order to realize a mode using
11 stacks, the XPath evaluating apparatus may be configured to
12 further include a stack generating unit for generating a
13 first stack which expresses the XPath with a string of stack
14 elements.

15 Another XPath evaluating apparatus according to the
16 present invention includes a document tree constructing unit
17 for inputting XML event strings which constitute an XML
18 document and serially constructing a document tree indicating
19 a document structure of the XML document based on inputted
20 XML events along with the input of the respective XML events,
21 an XML event transferring unit for inputting the XML event
22 strings which constitute the XML document to be processed and
23 serially transferring the XML event strings to the document
24 tree constructing unit, and an evaluation executing unit for

1 evaluating the XPath along with construction of the document
2 tree by this document tree constructing unit while using the
3 document tree with a part which has been constructed.

4 Furthermore, the present invention is also realized as
5 an information processing apparatus including an XML parser,
6 an XPath evaluating unit having a function of the above-
7 described XPath evaluating apparatus, and an application
8 executing unit for inputting an XML event generated by the
9 XML parser and executing processing of an XML document
10 configured by the inputted XML events in response to an
11 evaluation result of the XPath by the XPath evaluating unit.

12 Moreover, the present invention is also realized as a
13 program for controlling a computer to execute processing
14 corresponding to the respective steps of the above-described
15 XPath evaluating method, or as a program for causing a
16 computer to function as the XPath evaluating apparatus or the
17 information processing apparatus described above. This
18 program can be provided by storing and distributing the
19 program in a magnetic disk, an optical disk, a semiconductor
20 memory, and other recording media, or by means of distribut-
21 ing the program through a network.

22 Now, the present invention will be described in detail
23 based on example embodiments shown in the accompanying
24 drawings. The present invention performs the evaluation of

1 an XPath with respect to an XML document to be processed
2 which is inputted in accordance with a streaming format,
3 serially in response to the input of the XML document. As
4 embodiments for realizing the XPath evaluation by this
5 streaming processing, description will be made below regard-
6 ing an embodiment using an automaton (Embodiment 1), an
7 embodiment using stacks (Embodiment 2), and an embodiment
8 using a tree (Embodiment 3). Note that these embodiments are
9 generally realized by use of a personal computer, a worksta-
10 tion, and other computer apparatuses, or by use of various
11 information processing apparatus including a personal digital
12 assistant (PDA) or a cellular telephone.

13 **Embodiment 1**

14 In Embodiment 1, streaming processing of an XPath
15 evaluation is realized by use of an automaton. Figure 1 is a
16 view schematically showing an example of a hardware configu-
17 ration of an information processing apparatus which is
18 suitable for realizing this embodiment. An information
19 apparatus shown in Figure 1 includes: a central processing
20 unit (CPU) 101 which is operating means; a main memory 103
21 connected to the CPU 101 through a motherboard (M/B) chip set
22 102 and a CPU bus; a video card 104 similarly connected to
23 the CPU 101 through the M/B chip set 102 and an accelerated

1 graphics port (AGP); a hard disk 105, a network interface
2 106, and a USB port 107 which are connected to the M/B chip
3 set 102 through a peripheral component interconnect (PCI)
4 bus; and a floppy disk drive 109 and a keyboard/mouse 110
5 which are connected to the M/B chip set 102 via this PCI bus
6 through a bridge circuit 108 and a low-speed bus such as an
7 industry standard architecture (ISA) bus.

8 It is to be noted, however, that Figure 1 is just one
9 example of the hardware configuration of a computer for
10 realizing this embodiment. Accordingly, various other
11 configurations can be adopted as long as this embodiment is
12 applicable. For example, the computer may adopt a configura-
13 tion in which only a video memory is loaded instead of
14 providing the video card 104 while image data are processed
15 by the CPU 101. Alternatively, it is possible to provide a
16 drive for a compact disc read-only memory (CD-ROM) or a
17 digital versatile disc read-only memory (DVD-ROM) through an
18 inter face such as an AT attachment (ATA).

19 Figure 2 is a view showing an example of loading XPath
20 evaluating means (an analyzing system) in this embodiment.
21 As shown in Figure 2, an information processing apparatus
22 according to this embodiment includes an XML parser 10 for
23 parsing an XML document to be processed, an XPath evaluating
24 unit 20 for evaluating the XPath with respect to the parsed

1 XML document, and an application executing unit 30 for
2 executing given information processing by use of the XML
3 document after evaluation of the XPath.

4 The XML parser 10, the XPath evaluating unit 20, and
5 the application executing unit 30 shown in Figure 2 are
6 virtual software blocks which are realized by controlling the
7 CPU 101 with a program developed in the main memory 103 shown
8 in Figure 1, for example.

9 In Figure 2, the XML parser 10 inputs and parses the
10 XML document to be processed, and outputs and transmits an
11 XML event for notifying a result of analysis to the XPath
12 evaluating unit 20. The XML events are transmitted serially
13 to the XPath evaluating unit 20 depending on every node in a
14 tree structure (a document tree) of the XML document to be
15 analyzed. A publicly known XML parser, which has been
16 conventionally used in an information processing apparatus
17 dealing with XML documents, can be used as the XML parser 10.
18 In this embodiment, the XML document to be inputted is
19 assumed to be a well-formed document.

20 The XPath evaluating unit 20 receives the XML events
21 from the XML parser 10 and evaluates the XPath which is
22 provided in advance.

23 Figure 3 is a view showing a functional configuration
24 of the XPath evaluating unit 20.

1 As shown in Figure 3, the XPath evaluating unit 20 of
2 this embodiment includes an automaton generating unit 21, an
3 XML event transferring unit 22, and an evaluation executing
4 unit 23.

5 The automaton generating unit 21 generates an automaton
6 (a state transition machine) from the XPath which is provided
7 in advance. In this embodiment, the XPath is evaluated with
8 respect to an application using the API which is executed in
9 a streaming format such as the SAX or the XNI. Therefore,
10 due to the restriction by the streaming processing, functions
11 of the XPath to be evaluated are partially limited. For
12 example, an axis or a predicate which requires reference to a
13 previously notified XML event is not realized upon the
14 evaluation. However, such limitation does not cause a
15 problem in the course of actual processing in many
16 applications.

17 Now, limitation in the XPath will be concretely
18 described herein.

- 19 1. The XPath will be defined as a location path.
20 However, the use of an absolute path is permitted only
21 when the XPath is evaluated from a root node of a tree
22 structure of a subject XML document.

1 2. The axis should only be a forward axis. That is,
2 the axis will be any of the following:

3 self
4 child
5 descendant
6 descendant-or-self
7 following
8 following-sibling
9 attribute
10 namespace

11 Nevertheless, it is also possible to use a reverse axis under
12 a certain condition and on special loading. Details will be
13 described later.

14 3. The predicate will be treated as described below.

15 1) An operator will be any of the following:

16 or
17 and
18 =
19 !=

20 2) A primitive will be any of the following:

21 * a location path, provided that an axis thereof

1 will be any of the following:

2 attribute

3 namespace

4 self

5 parent

6 ancestor

7 ancestor-or-self;

8 * a literal;

9 * a number; and

10 * a function call, except any of the following:

11 last ()

12 id (object)

13 string (object?), which is set to a root

14 node or an element node.

15 Incidentally, in the XPath, a location path which does

16 not use an abbreviated notation is expressed by a row of an

17 axis, a node test, and a predicate, with junctions (which are

18 referred to as steps) sectioned by a symbol "/". The automa-

19 ton generating unit 21 expresses a state of evaluation of

20 each step with the automaton. This automaton performs the

21 following state transition.

22 Figure 4 is a view schematically showing an automaton

23 generated by the automaton generating unit 21. First, the

1 respective steps (the total quantity thereof is denoted by N)
2 are numbered starting from the left side (the head). A
3 situation where a step K is established at a time point when
4 a given token is read in will be defined as a state K. Here,
5 the situation where the step K is established refers to a
6 situation where a start tag token of a node satisfying the
7 step K has been read in but an end tag thereof has not been
8 read in yet. Meanwhile, an initial situation will be defined
9 as a state 0.

10 Here, an assumption is made herein that the start tag
11 token of the node satisfying the step K has been read in at
12 this moment so that a current situation moves to the state K.
13 When a node satisfying a state K+1 is read in this state K,
14 the situation moves to a state K+1. On the other hand, when
15 the end tag of the step K is read in the state K, the situa-
16 tion moves to a state K-1. While such state transition is
17 repeated, judgment is made that the location path is satis-
18 fied when the situation reaches a state N.

19 The above-described automaton generated by the automa-
20 ton generating unit 21 is retained in a work area of the main
21 memory 103 of Figure 1, for example. The automaton will be
22 used in processing by the evaluation executing unit 23 to be
23 described later.

24 The XML event transferring unit 22 inputs the XML

1 events outputted from the XML parser 10, and then serially
2 transfers the XML events to the evaluation executing unit 23
3 depending on the nodes in the tree structure of the XML
4 document. Moreover, the XML event transferring unit 22
5 serially transfers the XML events to the application execut-
6 ing unit 30 as well.

7 The evaluation executing unit 23 obtains the automaton
8 generated by the automaton generating unit 21 and receives
9 the XML events transferred from the XML event transferring
10 unit 22. Accordingly, the evaluation executing unit 23
11 performs transition between the states in the automaton while
12 serially reading token strings of the XML document. Thereaf-
13 ter, the evaluation executing unit 23 judges that the
14 location path is satisfied at a time point when a token
15 corresponding to the node satisfying the location path is
16 read in. Such an evaluation result is transmitted to the
17 application executing unit 30 as an evaluation event.
18 Concrete procedures of the evaluation processing by the
19 evaluation executing unit 23 will be described later.

20 The application executing unit 30 shown in Figure 2
21 inputs the XML events transferred from the XML event trans-
22 ferring unit 22 of the XPath evaluating unit 20 and the
23 evaluation event outputted from the evaluation executing unit
24 23, and executes processing of the XML document by streaming

1 processing.

2 Next, processing by the XPath evaluating unit 20 of
3 this embodiment will be described in more detail. As
4 described above, in this embodiment, the automaton expressing
5 the XPath to be evaluated is firstly generated and retained.
6 Then, the XML event strings constituting the XML document to
7 be processed are serially inputted, and the partial evalua-
8 tion of the XPath is performed by causing the states of the
9 automaton to perform transition based on these XML events.
10 The result of the partial evaluation of the XPath is retained
11 as the state of the automaton.

12 For the purpose of facilitation, the location path is
13 herein assumed not to be inclusive of the predicate, and only
14 "child" and "descendant" are considered as the axes. Treat-
15 ment of a step including any of "following-sibling", "follow-
16 ing", "self", "descendant-or-self", "attribute", and
17 "namespace" as the axis, or treatment of a step including a
18 predicate will be described later.

19 Figs. 5A to 5C are views showing examples of the
20 location path, the automaton, and the XML document to be
21 processed which are used in the processing according to this
22 embodiment. Given the location path shown in Figure 5A,
23 respective steps starting from the left in this location path
24 will be referred to as a step 1, a step 2, and a step 3,

1 respectively. The state of the automaton generated from this
2 location path by the automaton generating unit 21 will be as
3 shown in Figure 5B.

4 Here, an assumption will be made herein that the XML
5 document shown in Figure 5C is inputted. This XML document
6 is parsed by the XML parser 10 and sent to the XPath evaluat-
7 ing unit 20 as the XML events depending on the nodes. Then,
8 the XML event transferring unit 22 transfers the XML events
9 serially to the evaluation executing unit 23. Here, in
10 Figure 5C, an abbreviated notation <title/> is applied to a
11 case where there is no element content such as <title>
12 </title> (hereinafter similarly applicable to other XML
13 documents).

14 The evaluation executing unit 23 uses the above-
15 described automaton generated by the automaton generating
16 unit 21 and performs state transition of the automaton based
17 on the XML events transferred from the XML event transferring
18 unit 22, and thus evaluates the location path corresponding
19 to the automaton. Here, in the actual processing, the state
20 transition and the evaluation of the automaton are performed
21 by building a stack for analyzing a nested structure of the
22 XML document (an XML event string stack) in a memory area of
23 the main memory 103 of Figure 1., for example. That is, the
24 evaluation executing unit 23 retains the XML event strings

1 transmitted from the XML parser 10 serially in this XML event
2 string stack.

3 This XML event string stack pushes a node corresponding
4 to a start tag on the condition that the start tag is read
5 in, and pops the node on the condition that an end tag is
6 read in. A bottom of the stack will represent a root node
7 herein.

8 Accordingly, the evaluation executing unit 23 serially
9 performs pushing and popping to the XML event string stack in
10 accordance with the input of the XML event strings. Such
11 operations are performed consistently from the start to the
12 end of the XML document irrespective of whether the location
13 path is relative or absolute. During the operations, every
14 time when a given start tag is read in and a node correspond-
15 ing to the tag is pushed to the XML event string stack,
16 judgment is made as to whether or not the node (assuming the
17 current state as the state K) satisfies a step corresponding
18 to the state K+1 (i.e. the step K+1). If a result of this
19 judgment is "yes", the current state moves to the state K+1.
20 In this event, a link between a stack element and the state
21 is established. In other words, mapping is configured
22 between the pushed nodes and the states established by the
23 nodes. Regarding the state 0, a stack element corresponding
24 to a context node is linked when the location path is a

1 relative path; meanwhile, the bottom of the stack (i.e. the
2 root node) is linked when the location path is an absolute
3 path. On the other hand, in case of popping from the XML
4 event string stack, detection is made as to whether or not a
5 popped element is linked to the state. When the element is
6 linked to the state, the state is set back to the immediately
7 preceding state.

8 Now, description will be made sequentially with refer-
9 ence to Figs. 5A to 5C. First, at an initial state of the
10 state 0, a token <document> is read in and a node "document"
11 is pushed. In this event, the state of the XML event string
12 stack is set as:

13 bottom|document

14 Since this node "document" satisfies the step 1, the state
15 transition from the state 0 to the state 1 takes place.

16 Next, a token <title> is read in, and a node "title" is
17 pushed. In this event, the state of the XML event string
18 stack is set as:

19 bottom|document|title

20 This node "title" does not satisfy the condition

1 "child::chapter" of the node "document" linked to the state
2 1. That is, the node "title" does not satisfy the step 2.
3 Accordingly, no state transition takes place and reading a
4 token is continued instead.

5 After a token </title> being the end tag of the node
6 "title" is read in and the node "title" is popped, a token
7 <chapter> is subsequently read in and a node "chapter" is
8 pushed. Since this node satisfies the step 2, the state
9 transition from the state 1 to the state 2 takes place. In
10 this event, the state of the XML event string stack is set
11 as:

12 bottom|document|chapter

13 Next, a token </chapter> is read in and the node "chapter" is
14 thereby popped. In this event, the state 2 is linked to the
15 stack element to be popped. Therefore, in this case, the
16 state of the automaton is transferred from the state 2 back
17 to the immediately preceding state 1 simultaneously with the
18 popping.

19 Thereafter, the token <chapter> is read in again and
20 the node "chapter" is pushed, whereby the state transition
21 from the state 1 to the state 2 takes place. Moreover, a

1 token <section> is read in and a node "section" is pushed.
2 Since this node satisfies the step 3, the state transition
3 from the state 2 to the state 3 takes place. At this time
4 point, the evaluation executing unit 23 judges that the
5 location path is established, and transmits the evaluation
6 event indicating the evaluation result to the application
7 executing unit 30.

8 In the foregoing description, the evaluation executing
9 unit 23 is designed to output the evaluation event when the
10 location path is established. However; it is also possible
11 to design the evaluation executing unit 23 to output an
12 evaluation event indicating establishment or
13 non-establishment every time when each XML event is
14 processed.

15 Next, a description will be made regarding treatment of
16 other forward axes.

17 1. Treatment of "attribute" and "namespace"

18 To evaluate a location path including a step with an
19 axis "attribute", a step corresponding to a state immediately
20 subsequent to a state after transition is detected upon state
21 transition. Then, if the step is "attribute", the step is
22 evaluated by use of an attribute of the most recently input-
23 ted token. The state is moved to one notch forward when a

1 node satisfies the step in this evaluation. On the contrary,
2 if the node does not satisfy the step, the original state
3 transition is canceled.

4 Regarding "namespace", the "namespace" in the current
5 context is managed, and then the evaluation of the step is
6 executed similarly to the case of "attribute".

7 Concerning "attribute" and "namespace", an operation of
8 the XML event string stack does not take place in the process
9 when any of these axes is established. Therefore, a node to
10 be linked to any of those states is deemed as a NULL node,
11 which represents that no node is applicable thereto.

12 2. Treatment of "self"

13 To evaluate a location path including a step with an
14 axis "self", a step corresponding to a state immediately
15 subsequent to a state after transition is detected upon state
16 transition. Then, the step is evaluated if the step is
17 "self". The state moves to one notch forward when a node
18 satisfies the step in this evaluation. On the contrary, if
19 the node does not satisfy the step, the original state
20 transition is canceled. An operation of the XML event string
21 stack does not take place in the process when the axis "self"
22 is established. Therefore, a node to be linked to this state
23 is deemed as a NULL node, which represents that no node is

1 applicable thereto.

2 3. Treatment of "descendant-or-self"

3 To evaluate a location path including a step with an
4 axis "descendant-or-self", the axis of the step is firstly
5 regarded as "self", and the operation in the above-described
6 section 2 is performed. When judgment is made in this opera-
7 tion that a node does not satisfy the step, the evaluation is
8 continued while the axis is regarded as "descendant".

9 4. Treatment of "following-sibling"

10 The following operation is performed to evaluate a
11 location path including a step with an axis
12 "following-sibling".

13 To begin with, a step with an axis "following-sibling"
14 is defined as a step K and a state corresponding thereto is
15 defined as a state K. Basically, with respect to a stack top
16 node at a time point when a node which was pushed upon estab-
17 lishment of a step K-1 is popped, a node which will be pushed
18 subsequent to the node will be a node to be selected in the
19 step K. However, attention is required herein.

20 Specifically, the node which satisfied this step K is not
21 only a candidate for the node to be selected in the step K
22 but also a candidate to be selected in the step K-1 at the

1 same time. For example, a location path is herein assumed to
2 be described as follows:

3 /child::document/child::chapter/following-sibling::chapter

4 In this case, the second "chapter" in the following descrip-
5 tion can be selected both in the second step and in the third
6 step:

```
7      <document>
8          <chapter>--1
9          </chapter>
10         <chapter>--2
11         </chapter>
12         <chapter>--3
13         </chapter>
14     </document>
```

15 Such ambiguity must be strictly interpreted in the
16 event of performing node selection depending on "position".
17 For example, the second chapter and the third chapter must be
18 selected at the same time according to the location path
19 described as follows:

1 /child::document/child::chapter/following-sibling::chapter[1]

2 Meanwhile, the third chapter must be selected according to
3 the location path described as follows:

4 /child::document/child::chapter[2]/following-sibling::chapter

5 Accordingly, in this embodiment, the process is divided
6 in two at a time point when the node is judged to satisfy the
7 step K-1, and processing is performed such that the respec-
8 tive processes continue state transition corresponding to two
9 ways of interpretation of the location path. In one of the
10 processes, the state performs transition to the state which
11 is immediately precedent to the state linked to the node at
12 this point (which is normally a state K-2, or a state immedi-
13 ately precedent thereto when the state K-2 includes "self"),
14 and the evaluation is continued thereafter.

15 The following operation takes place in the other
16 process. Firstly, the state remains at the state K-1. In
17 the state K-1, if any node is pushed while the stack top node
18 at the time point when the node pushed as a result of a
19 judgment to satisfy the step K-1 is popped remains as a stack
20 top, then judgment is made whether or not the node satisfies
21 the step K. When the node satisfies the step K, the state is

1 transferred to the state K. Then, the pushed node is linked
2 to the state K and the evaluation is continued thereafter.
3 Meanwhile, when the node linked to the state K is popped, the
4 state is set back to the state K-1. In the state K-1, when
5 an end tag token for the node satisfying the step K-1 is
6 inputted, that is, when the stack top node at the time point
7 when the node pushed as a result of the judgment to satisfy
8 the step K-1 is popped is popped, the operation is terminated
9 because there is no more node to be selected in the step K in
10 this process.

11 5. Treatment of "following"

12 The following operation is performed to evaluate a
13 location path including a step with an axis "following".

14 Since this case also bears a possibility to cause
15 ambiguity of the location path, division of the process
16 similar to the case of "following-sibling" is required. To
17 begin with, a step with an axis "following" is defined as a
18 step K and a state corresponding thereto is defined as a
19 state K. Similar to the above-described section 4, the
20 process is divided in two at a time point when the node is
21 judged to satisfy the step K-1. In one of the processes, the
22 state performs transition to the state which is immediately
23 precedent to the state linked to the node at this point, and

1 the evaluation is continued thereafter.

2 The following operation takes place in the other
3 process. Firstly, the state remains at the state K-1. If
4 any node is pushed in the state K-1, then a judgment is made
5 whether or not the node satisfies the step K. When the node
6 satisfies the step K, the state is transferred to the state
7 K. Then, the pushed node is linked to the state K and the
8 evaluation is continued thereafter. In the case of "follow-
9 ing", the state remains as the state K even when an end tag
10 token for the node satisfying the step K-1 is inputted in the
11 state K-1.

12 Next, a description will be made regarding treatment of
13 predicates.

14 Basically, a predicate can be treated when it is possi-
15 ble to compose the predicate by use of information accumu-
16 lated in the stack at the moment of state transition. That
17 is, when a start tag token satisfying the step K is read in,
18 it is possible to describe information regarding the node
19 which can be composed in that state or information regarding
20 an "ancestor" of the node remaining in the stack as the
21 predicate. Moreover, it is possible to treat a position
22 function by providing counters to the respective states. In
23 the state of establishment of the step as described above,
24 one notch is added to the counter of the state corresponding

1 to the step. The predicate including the position function
2 is evaluated based on the value of this counter, and a
3 judgment is made whether or not the node satisfies the step
4 eventually.

5 Next, description will be made regarding treatment of
6 reverse axes.

7 In this embodiment, it is possible to use a reverse
8 axis such as "parent", "ancestor", or "ancestor-or-self" in
9 very limited location paths. As described in the treatment
10 of the predicates, the information on the ancestor of the
11 current node is accumulated as the state of the automaton.
12 Therefore, if all the steps on the right side from a certain
13 point of a location path are any of "parent", "ancestor",
14 "ancestor-or-self" and "self", then a stack operation may be
15 stopped at that point and the steps may be evaluated while
16 retracing those steps.

17 Figure 6 is a flowchart which explains streaming
18 processing of an XML document according to this embodiment.
19 As shown in Figure 6, the XPath evaluating unit 20 generates
20 an automaton from the location path of the XPath provided in
21 advance by the automaton generating unit 21 as an initial
22 operation (Step 601). When an XML document to be processed
23 is inputted and XML events are sent from the XML parser 10
24 (Step 602), the evaluation executing unit 23 of the XPath

1 evaluating unit 20 serially judges the state transition in
2 the automaton corresponding to the XML events (Step 603).
3 This state transition represents the partial evaluation of
4 the location path (the XPath) corresponding to the automaton.
5 The result of this partial evaluation is retained as the
6 current state of the automaton.

7 Thereafter, the input of the XML events and judgment of
8 the state transition are repeated until the automaton reaches
9 the final state. When the automaton reaches the final state,
10 the location path corresponding to this XML document is
11 evaluated as established, and the evaluation event is sent to
12 the application executing unit 30 (Steps 604 and 605).

13 The application executing unit 30 obtains the XML
14 events outputted from the XML parser 10 and the evaluation
15 events outputted from the evaluation executing unit 23 of the
16 XPath evaluating unit 20. The application executing unit 30
17 serially processes the XML events in accordance with the
18 evaluation event.

19 When all the processing from Steps 602 to 605 is
20 executed for all the XML events regarding the XML document to
21 be processed which are sent from the XML parser 10, the
22 streaming processing of the XML document according to this
23 embodiment is completed (Step 606).

1 **Embodiment 2**

2 In Embodiment 2, the evaluation of the XPath corre-
3 sponding to the processing by use of the state transition of
4 the automaton as described in Embodiment 1 is realized by use
5 of stacks for analyzing a nested structure of an XML
6 document. That is, this embodiment is implemented by a
7 method of expressing both of the XPath and XML event strings
8 with stacks and comparing the XPath and the XML events every
9 time the XML event is notified. Similar to Embodiment 1,
10 this embodiment is realized by , for example, the information
11 processing apparatus shown in Figure 1.

12 Figure 7 is a view showing an example of implementing
13 XPath evaluating means (an analyzing system) in this embodi-
14 ment. As shown in Figure 7, the information processing
15 apparatus according to this embodiment includes an XML parser
16 10 for parsing an XML document to be processed, an XPath
17 evaluating unit 40 for evaluating the XPath with respect to
18 the parsed XML document, and an application executing unit 30
19 for executing given information processing by use of the XML
20 document after the evaluation of the XPath.

21 The XML parser 10, the XPath evaluating unit 40, and
22 the application executing unit 30 shown in Figure 7 are
23 virtual software blocks which are realized by controlling the
24 CPU 101 with a program developed in the main memory 103 shown

1 in Figure 1, for example. Since the XML parser 10 and the
2 application executing unit 30 are similar to the correspond-
3 ing constituents in Embodiment 1, same reference numerals are
4 designated and explanation thereof is omitted.

5 The XPath evaluating unit 40 receives the XML events
6 from the XML parser 10 and evaluates the XPath which has been
7 provided in advance.

8 Figure 8 is a view showing a functional configuration
9 of the XPath evaluating unit 40. As shown in Figure 8, the
10 XPath evaluating unit 40 of this embodiment includes a stack
11 generating unit 41, an XML event transferring unit 42, and an
12 evaluation executing unit 43.

13 The stack generating unit 41 generates a stack from the
14 XPath which has been provided in advance. Here, the stack is
15 a data structure for expressing a step of the XPath. Genera-
16 tion of the stack from the XPath is performed as follows.

17 Specifically, regarding the respective steps of the
18 XPath, the stack generating unit 41 expresses a node test and
19 a predicate by stack elements and pushes the stack elements
20 to the stack in accordance with an axis. For example,
21 regarding "child::para", "para" is expressed by a stack
22 element and is pushed to an existing stack. Here, the node
23 test and the predicate may be expressed by any kinds of stack
24 elements as long as the node test and the predicate can be

1 evaluated appropriately with respect to the XML event
2 strings. However, when a predicate "position ()" is included
3 therein, a counter is prepared for retaining a frequency of
4 coincidence of the stack.

5 Figure 9 is a view exemplifying relations between the
6 types of axes and the stacks to be generated. Moreover, in
7 Figure 9, respective arrows (straight arrows and waved
8 arrows) represent comparison of an original stack and
9 comparison of a subsequent stack in case of coincidence. To
10 be more precise, a straight arrow represents comparison of an
11 original stack and comparison of a subsequent stack in case
12 of coincidence. On the contrary, a waved arrow represents
13 comparison of an original stack and comparison of a subse-
14 quent stack in case of coincidence and only when a specific
15 XML event is notified. The specific XML event in this case
16 varies depending on the type of the axis (a difference
17 between "following-sibling" and "following").

18 * In case of "following-sibling"

19 XML events corresponding to sibling nodes and descen-
20 dant nodes thereof after a node coincident with the stack
21 element at the head of the original stack. A subsequent
22 stack is compared based on a stack element corresponding to
23 the XML event relevant to the sibling node.

24 * In case of "following"

1 XML events corresponding to nodes after a node coinci-
2 dent with the stack element at the head of the original
3 stack. However, XML events corresponding to descendant nodes
4 thereof are excluded. A subsequent stack is compared based
5 on stack elements corresponding to the XML events respec-
6 tively relevant to the nodes described above.

7 Figure 10 is a view showing a variety of stacks to be
8 generated by the XPath based on the relations shown in Figure
9 9. The stacks shown in Figure 10 are respectively generated
10 from each of the XPath of:

- 11 1) chapter/para;
- 12 2) chapter/para[@type="warning"];
- 13 3) chapter/para[2]; and
- 14 4) ../para

15 Here, in the description above and in Figure 10, description
16 of a "child" axis is omitted based on the rules of notation
17 in the XPath.

18 The stack generated by the stack generating unit 41 as
19 described above is retained in the work area of the main
20 memory 103 of Figure 1, for example, and are used in the
21 processing by the evaluation executing unit 43 to be
22 described later.

1 Note that the limitation in the functions of the XPath
2 is applicable in this embodiment similar to Embodiment 1.

3 The XML event transferring unit 42 inputs the XML
4 events outputted from the XML parser 10, and then serially
5 transfers the XML events to the evaluation executing unit 43
6 depending on the nodes in the tree structure of the XML
7 document. Moreover, the XML event transferring unit 42
8 serially transfers the XML events to the application execut-
9 ing unit 30 as well.

10 The evaluation executing unit 43 obtains the stack
11 generated by the stack generating unit 41 and receives the
12 XML events transferred from the XML event transferring unit
13 42. Accordingly, the evaluation executing unit 43 operates
14 the stack while serially reading token strings of the XML
15 document. Thereafter, the evaluation executing unit 43
16 judges that the XPath is satisfied at a time point when a
17 token corresponding to the node satisfying the XPath is read
18 in. Such an evaluation result is transmitted to the applica-
19 tion executing unit 30 as an evaluation event.

20 In this embodiment, similar to Embodiment 1, a stack
21 for analyzing a nested structure of an XML document (an XML
22 event string stack) is introduced as a data structure for
23 realizing evaluation of the XPath by use of the above-
24 described stack. That is, the evaluation executing unit 43

1 prepares the XML event string stack in addition to the above-
2 described stack for the XPath generated by the stack generat-
3 ing unit 41, and retains the XML event strings transmitted
4 from the XML parser 10 in this XML event string stack.

5 Regarding this XML event string stack, when an XML event
6 corresponding to a start tag is inputted, the XML event is
7 expressed by a stack element and is pushed to the stack in
8 principle. Moreover, the stack is popped when an XML event
9 corresponding to an end tag is inputted. Meanwhile, when an
10 XML event corresponding to a text or the like is inputted,
11 this XML event is also expressed by a stack element and is
12 pushed to the stack. However, this stack is popped immedi-
13 ately after comparison of the stack is completed. Here, the
14 XML event may be expressed by any stack element as long as
15 the information retained by the event continues to be
16 retained.

17 Moreover, the evaluation executing unit 43 evaluates
18 the XPath every time when the XML event is transferred from
19 the XML event transferring unit 42 by comparing the XML event
20 string stack and the stack for the XPath based on a stack
21 element corresponding to an XML event of a noted node in the
22 XML document. Here, comparison of stacks refers to compari-
23 son of corresponding stack elements. Meanwhile, comparison
24 of the stack elements refers to evaluating a node test and

1 predicate which correspond to the relevant XML event (a
2 previously notified XML event).

3 Comparison of the stacks is started from the final stack
4 element in principle. When a mismatched stack element is
5 detected, the comparison processing is terminated
6 immediately. Moreover, it is possible to compare all the
7 stack elements every time, or alternatively, it is possible
8 to retain the position of the stack element which was
9 mismatched upon comparison and to resume comparison from the
10 mismatched stack element in the next comparison processing.

11 Coincidence of the stack means matching of the XPath
12 with the node corresponding to the event which is notified
13 most recently. However, when the counter is provided to the
14 stack for the XPath, a count value of the corresponding stack
15 element in the XML event string stack is incremented every
16 time of coincidence of the stack. Then, a judgment is made
17 that the XPath is matched only when the count value of the
18 stack element coincides with a count value of the stack for
19 the XPath.

20 As described above, in this embodiment, the stack for
21 the XPath (a first stack) is generated in which the XPath to
22 be evaluated is expressed by a string of stack elements, and
23 the XML event string stack (a second stack) is generated for
24 analyzing the nested structure of the XML document to be

1 processed based on each of the XML events inputted by input-
2 ting the XML event string which constitutes the XML document
3 to be processed. Thereafter, partial evaluation of the XPath
4 is performed by comparing the first stack with the second
5 stack. A result of the partial evaluation of the XPath is
6 retained as a stack element of the XML event string stack.

7 Moreover, as described above, operation of the XML
8 event string stack is similar to the operation of the XML
9 event string stack described in Embodiment 1. Therefore, the
10 operation and the comparison of the XML event string stack
11 correspond to the state transition of the automaton described
12 in Embodiment 1. However, implementation is easy in Embodi-
13 ment 2 because the evaluation of the XPath is executed only
14 by matching the data accumulated in the stack.

15 Next, a description will be made in more detail regard-
16 ing the processing by the XPath evaluating unit 40 of this
17 embodiment.

18 Figure 11 is a view showing an example of the XML
19 document to be processed, and Figure 12A to Figure 14B are
20 views showing aspects of the evaluation processing using the
21 stack with respect to the XML document of Figure 11. Here,
22 in Figure 11, abbreviated notations such as abbreviating
23 "document" to "doc" are applied to respective tags as appro-
24 priate (hereinafter similarly applicable to other XML

1 documents).

2 An assumption is made herein that "chapter/para[2]"
3 shown in (3) of Figure 10 is given as the XPath. That is,
4 establishment of this XPath is judged when a node "para" is
5 detected two notches behind a node "chapter" in a given event
6 string of the XML document. The stack to be generated by the
7 stack generating unit 41 with respect to the XPath is as
8 shown in Figure 10.

9 Figure 12A to Figure 14B shows SAX event strings (the
10 XML event strings) with respect to the XML document of Figure
11 11 and aspects of comparison by the evaluation executing unit
12 43 when the respective SAX event strings are inputted in a
13 sequential order.

14 To begin with, a SAX event "startDocument" indicating a
15 start of a document is inputted, and comparison of the stacks
16 is initiated (Figure 12A). At this point, the stack element
17 of the XML event string stack is empty (NULL). Next, a SAX
18 event "startElement: doc" indicating a start of an element (a
19 node) is inputted (Figure 12B), and reading of a token (a SAX
20 event) is continued further.

21 Next, a SAX event "startElement: chapter" indicating a
22 start of the element is inputted, and the XML event string
23 stack is pushed and compared with the stack for the XPath
24 (Fig, 12C). Although this stack element is matched, the

1 stack element "para" is not matched at this point of time.
2 Accordingly, reading of a token (a SAX event) is continued
3 further. Subsequently, a SAX event "startElement: para" is
4 inputted, and the XML event string stack element is pushed
5 and compared with the stack for the XPath (Figure 13A).
6 Although this stack is matched, since this is the first node
7 "para" appearing in the XML document shown in Figure 11, a
8 counter value (current_pos) for dealing with a position
9 function (pos=2) is incremented by one notch, and then
10 reading of a token (a SAX event) is continued further.

11 Thereafter, SAX events "endElement: para" and "endEle-
12 ment: chapter" indicating ends of the elements are inputted
13 and the corresponding stack elements in the XML event string
14 stack are respectively popped (Figs. 13B and 13C). Then, the
15 SAX event "startElement: chapter" is inputted again, and the
16 XML event string stack is pushed and compared with the stack
17 for the XPath (Figure 14A). This stack element is matched.
18 Subsequently, the SAX event "startElement: para" is inputted,
19 and the XML event string stack is pushed and compared with
20 the stack for the XPath (Figure 14B). This stack element is
21 matched and the counter value is incremented by one notch.
22 Accordingly, the counter value becomes "current_pos=2" and
23 thereby satisfies a demand of the position function. There-
24 fore, a judgment is made that the XPath "chapter/para[2]" is

1 established in the XML document of Figure 11.

2 Incidentally, in the foregoing description, the node is
3 a candidate for the node to be selected in the step K and
4 also a candidate for the node to be selected in the step K-1
5 at the same time, concerning "following-sibling" and "follow-
6 ing". Accordingly, in this embodiment, the process is
7 divided in two at the time point when the node pushed upon
8 establishment of the step K-1 is popped, and the respective
9 processes continue stack operations corresponding to two ways
10 of interpretation of the location path. Such operations will
11 be described in detail with concrete examples.

12 Figure 15 is a view showing an example of the XML
13 document to be processed, Figure 16 is a view showing the
14 XPath to be evaluated and the stack to be generated from this
15 XPath. Figure 17A to Figure 19B are views showing aspects of
16 the evaluation processing using the stack with respect to the
17 XML document of Figure 15.

18 As shown in Figure 16, an assumption is made herein
19 that "chap/para[@num="2"]/following-sibling::para" is given
20 as the XPath. That is, establishment of this XPath is evalu-
21 ated when a sibling node "para" is detected subsequent to a
22 node "para num="2"" behind a node "chapter" in a given event
23 string of the XML document.

24 Figure 17A to Figure 19B, in a sequential order, shows

1 SAX event strings (the XML event strings) with respect to the
2 XML document of Figure 15 and aspects of comparison by the
3 evaluation executing unit 43 when the respective SAX event
4 strings are inputted.

5 To begin with, a SAX event "startDocument" indicating a
6 start of a document, and subsequently a SAX event "startEle-
7 ment: doc" indicating a start of an element are inputted
8 (Figs. 17A and 17B), and reading of a token (a SAX event) is
9 continued further. Next, a SAX event "startElement: chapter"
10 indicating a start of the element is inputted, and the XML
11 event string stack is pushed and compared with the stack for
12 the XPath (Figure 17C). Since the stack element "para" is
13 not matched at this time point,
14 reading of a token (a SAX event) is continued further.

15 Next, a SAX event "startElement: para, @num=1" is
16 inputted, and the XML event string stack is pushed and
17 compared with the stack for the XPath (Figure 18A). This
18 stack is not matched with the corresponding stack element
19 (para@num=2) of the stack for the XPath. Accordingly,
20 reading of a token (a SAX event) is continued further.
21 Subsequently, a SAX event "endElement: para" indicating an
22 end of the element is inputted and the corresponding stack
23 elements in the XML event string stack are popped respec-
24 tively (Figure 18B). Thereafter, a SAX event "startElement:

1 para, @num=2" is inputted, and the XML event string stack is
2 pushed and compared with the stack for the XPath (Figure
3 18C). This stack element is matched with the corresponding
4 stack element of the stack for the XPath. Therefore, the
5 comparison with the last half of the stack for the XPath (the
6 stack element "para") shown in Figure 16 is performed from
7 that point on.

8 Next, the SAX event "endElement: para" is inputted, and
9 the corresponding stack elements in the XML event string
10 stack are popped respectively (Figure 19A). Thereafter, a
11 SAX event "startElement: para, @num=3" is inputted, and the
12 XML event string stack is pushed and compared with the stack
13 for the XPath (Figure 19B). This stack element is matched
14 with the last half of the stack for the XPath (the stack
15 element "para"). Therefore, a judgment is made that the
16 XPath "chap/para[@num="2"]/following-sibling::para" is estab-
17 lished in the XML document of Figure 15.

18 Figure 20 is a flowchart which explains streaming
19 processing of an XML document according to this embodiment.

20 As shown in Figure 20, the XPath evaluating unit 40
21 generates the stack for the XPath from the location path of
22 the XPath provided in advance by use of the stack generating
23 unit 41 as an initial operation (Step 2001). When the XML
24 document to be processed is inputted and the XML events are

1 sent from the XML parser 10 (Step 2002), the evaluation
2 executing unit 43 of the XPath evaluating unit 40 generates
3 the XML event string stack as described above, and the
4 comparison with the stack for the XPath is serially judged
5 (Step 2003). The result of this comparison represents the
6 partial evaluation of the XPath expressed by the stack for
7 the XPath.

8 Thereafter, the input of the XML events and comparison
9 of the stacks are repeated until this XML event string stack
10 and the stack for the XPath are entirely matched. When both
11 of the stacks are entirely matched, the location path corre-
12 sponding to this XML document is judged as established, and
13 the evaluation event is sent to the application executing
14 unit 30 (Steps 2004 and 2005).

15 The application executing unit 30 obtains the XML
16 events outputted from the XML parser 10 and the evaluation
17 event outputted from the evaluation executing unit 43 of the
18 XPath evaluating unit 40. The application executing unit 30
19 serially processes the XML events in accordance with the
20 evaluation event.

21 When all the processing from Steps 2002 to 2005 is
22 executed for all the XML events regarding the XML document to
23 be processed which are sent from the XML parser 10, the
24 streaming processing of the XML document according to this

1 embodiment is completed (Step 2006).

2 **Embodiment 3**

3 In Embodiment 3, a tree with respect to an XML document
4 (a document tree) is serially constructed while inputting XML
5 event strings in a streaming format, and evaluation of the
6 XPath is sequentially performed along with construction of
7 this document tree.

8 As described above, in Embodiment 1, the automaton is
9 generated from the XPath to be evaluated and the XPath is
10 evaluated by serially performing the state transition of the
11 automaton in accordance with the XML event strings which are
12 inputted in the streaming format. Meanwhile, in Embodiment
13 2, the steps of the XPath are expressed by the first stack
14 and the nested structure of the XML document inputted in the
15 streaming format is expressed by the second stack, and the
16 XPath is evaluated by comparing the contents of these stacks.
17 Each of these methods loses the information regarding the
18 evaluated portion of the XML event strings. Accordingly, it
19 is possible to perform evaluation concerning a reverse axis
20 under a very limited condition.

21 On the contrary, in this embodiment, the document tree
22 is constructed along with the input of the XML event strings.
23 Accordingly, all the information before inputting a given XML

1 event is retained as the tree structure. Therefore, it
2 becomes possible to evaluate the XPath including a step based
3 on a reverse axis without any problem. Similar to Embodiment
4 1, this embodiment is realized by the information processing
5 apparatus shown in Figure 1, for example.

6 Figure 21 is a view showing an example of loading XPath
7 evaluating means (an analyzing system) in this embodiment.
8 As shown in Figure 21, the information processing apparatus
9 according to this embodiment includes an XML parser 10 for
10 parsing an XML document to be processed, an XPath evaluating
11 unit 50 for evaluating the XPath with respect to the parsed
12 XML document, and an application executing unit 30 for
13 executing given information processing by use of the XML
14 document after the evaluation of the XPath.

15 The XML parser 10, the XPath evaluating unit 50, and
16 the application executing unit 30 shown in Figure 21 are
17 virtual software blocks which are realized by controlling the
18 CPU 101 with a program developed in the main memory 103 shown
19 in Figure 1, for example. Since the XML parser 10 and the
20 application executing unit 30 are similar to the correspond-
21 ing constituents in Embodiment 1, same reference numerals are
22 designated and explanation thereof is omitted.

23 The XPath evaluating unit 50 receives the XML events
24 from the XML parser 10 and evaluates the XPath which has been

1 provided in advance.

2 Figure 22 is a view showing a functional configuration
3 of the XPath evaluating unit 50. As shown in Figure 22, the
4 XPath evaluating unit 50 of this embodiment includes an XPath
5 storage unit 51, an XML event transferring unit 52, a
6 document tree constructing unit 53, and an evaluation execut-
7 ing unit 54.

8 The XPath storage unit 51 is realized by the main
9 memory 103 of the computer apparatus shown in Figure 1, for
10 example. The XPath storage unit 51 retains the XPath which
11 has been provided in advance. The XML event transferring
12 unit 52 inputs the XML events outputted from the XML parser
13 10, and then serially sends the XML events to the document
14 tree constructing unit 53 depending on the nodes in the tree
15 structure of the XML document. Moreover, the XML event
16 transferring unit 52 serially transfers the XML events to the
17 application executing unit 30 as well.

18 The document tree constructing unit 53 receives the XML
19 events transferred from the XML event transferring unit 52,
20 and constructs a document tree corresponding to the XML
21 events which are transferred at the point of time. This
22 document tree reflects the structure of the XML events
23 obtained at the point of time. Accordingly, the document
24 tree constitutes a subtree with respect to the document tree

1 representing the structure of the entire XML document to be
2 processed. Meanwhile, the document tree constructed in the
3 event of obtaining the XML event strings to the end is
4 identical to the document tree of the entire XML document.

5 Construction of the document tree is performed by
6 adding a new node when a start tag token corresponding to an
7 element of the XML document is inputted as an XML event.
8 Moreover, when the node is added as a result of input of the
9 start tag token, the node is regarded as an inserting
10 position thereafter. Meanwhile, when an end tag token for
11 the relevant element is inputted, a parent node of the node
12 is regarded as the inserting position thereafter. Therefore,
13 this document tree is updated every time when the document
14 tree constructing unit 53 inputs an XML event.

15 The document tree which is constructed (updated) by the
16 document tree constructing unit 53 is retained in the work
17 area of the main memory 103 of Figure 1, for example, and is
18 used in the processing by the evaluation executing unit 54 to
19 be described later.

20 When the above-described document tree is updated by
21 the document tree constructing unit 53, the evaluation
22 executing unit 54 evaluates the XPath retained in the XPath
23 storage unit 51 by use of the document tree. Regarding a
24 step evaluated as established as a result of evaluation of

1 the XPath, the evaluation executing unit 54 of this embodi-
2 ment saves information on a node of the document tree
3 selected in the relevant step, that is, information on a
4 result of partial evaluation of the XPath. This information
5 is expressed as a set of tuples of a step ID and a node ID of
6 each step {<sid1, nid1>, <sid1, nid2>, ...}. Such a set is
7 referred to as a saved set. The saved set thus generated is
8 retained in the work area of the main memory 103 of Figure 1,
9 for example.

10 In the saved set, the respective steps are numbered in
11 order starting from number 1. Here, the 0th step is defined
12 as a special step, and a root node is always saved so as to
13 correspond to this step. Each node selected owing to the
14 evaluation by the evaluation executing unit 54 is saved as a
15 context node unless a node set selected in each subsequent
16 step becomes empty.

17 Figure 23 to Figure 25 are flowcharts showing proce-
18 dures for evaluating the XPath by the evaluation executing
19 unit 54 using the document tree and the saved set. Figs. 23
20 and 24 show an operation when a start tag token is inputted
21 as the XML event, and Figure 25 shows an operation when an
22 end tag token is inputted as the XML event. Here, for the
23 purpose of facilitating explanation, the illustrated opera-
24 tions do not consider a predicate. Response to a predicate

1 will be described later.

2 As shown in Figure 23, when the start tag token is
3 inputted as an XML event (Step 2301), the evaluation execut-
4 ing unit 54 inserts a node N corresponding to this element to
5 a document tree which is constructed at the point of time
6 (Step 2302). Initially, the node N is inserted with respect
7 to the root node. Then regarding each element <S, nid> in
8 the saved set, the following operation is performed with
9 reference to a step S' which is numbered immediately subse-
10 quent to a step S (Step 2303).

11 The evaluation executing unit 54 checks whether the
12 step S' is the last step of the location path (XPath) (Step
13 2304). When the Step S' is the last step, the evaluation
14 executing unit 54 checks whether the evaluation of the step
15 S' is established or not (Step 2305). If the evaluation is
16 established, then the node N which was inserted lastly is the
17 node addressed by the XPath. Accordingly, this node N is
18 outputted as an evaluation result (Step 2306).

19 When the step S' is not the last step of the location
20 path, the evaluation executing unit 54 checks whether the
21 evaluation of the step S' is established or not (Step 2307).
22 When the evaluation is established, the evaluation executing
23 unit 54 further performs the following processing in response
24 to an axis in a step S" which is subsequent to the step S'.

1 Note that when the evaluation of the step S' is not
2 established by the judgments made in steps 2305 to 2307, the
3 processing regarding the XML event is terminated and input of
4 the next XML event is waited for.

5 As shown in Figure 24, when the axis of the step S" is
6 a "self" axis, the processing by the evaluation executing
7 unit 54 returns to the judgment in Step 2304 while setting
8 the step S" as the new step S' (Steps 2308 and 2309).

9 When the axis of the step S" is a forward axis such as
10 "child" or "descendant", a judgment is made first as to
11 whether there is a possibility of later input which satisfies
12 a relation designated by the axis of the step S" while
13 setting the node N as the context node (Steps 2308 and 2310).
14 If there is a possibility of such input, a tuple (a set of
15 the step ID and the node ID) <S', N> is added to the saved
16 set (Step 2311). Then, each element in a node set positioned
17 in the document order subsequent to the node N (inclusive of
18 the node N in the case of the axis containing "self" (such as
19 descendant-or-self)) in the document tree constructed at the
20 point of time is set as the new node N, and the step S" is
21 set as the new step S'. Then, the processing by the evalua-
22 tion executing unit 54 returns to the judgment in Step 2304
23 (Step 2312).

24 When there is no possibility of input which satisfies

1 the relation designated by the axis of the step S", each
2 element in a node set positioned in the document order subse-
3 quent to the node N (inclusive of the node N in the case of
4 the axis containing "self" (such as "descendant-or-self")) in
5 the document tree constructed at the point of time is set as
6 the new node N, and the step S" is set as the new step S'
7 without adding the tuple <S', N> to the saved set. Then, the
8 processing by the evaluation executing unit 54 returns to the
9 judgment in Step 2304 (Steps 2310 and 2312).

10 When the axis of the step S" is a reverse axis such as
11 "parent" or "ancestor", each element in a node set positioned
12 in the document subsequent order to the node N (inclusive of
13 the node N in the case of the axis containing "self" (such as
14 "ancestor-or-self")) in the document tree constructed at the
15 point of time is set as the new node N, and the step S" is
16 set as the new step S'. Then, the processing by the evalua-
17 tion executing unit 54 returns to the judgment in Step 2304
18 (Steps 2308 and 2312).

19 In the foregoing description, the evaluation executing
20 unit 54 is designed to output the evaluation result when the
21 XPath is established (see Step 2306). However, similar to
22 Embodiments 1 and 2, it is also possible to design the
23 evaluation executing unit 54 to output an evaluation result
24 indicating establishment or non-establishment every time when

1 each XML event is processed.

2 As shown in Figure 25, when the end tag token is input-
3 ted as the XML event (Step 2501), the evaluation executing
4 unit 54 detects a tuple <S, N> containing a node correspond-
5 ing to the token out of the saved set (Step 2502). Next,
6 concerning the detected tuple, a judgment is made as to
7 whether there is a possibility of later input which satisfies
8 a relation designated by the axis of the step S', which is
9 numbered immediately subsequent to the step S, while setting
10 the node N as the context node (Steps 2503). Then, the tuple
11 without possibility of such input is removed from the saved
12 set (Step 2504).

13 Thereafter, a tuple <S', N'> containing a child node of
14 the token to be processed is detected out of the saved set
15 (Step 2505). Next, concerning the detected tuple, a judgment
16 is made as to whether there is a possibility of later input
17 which satisfies a relation designated by the axis of the step
18 S'', which is numbered immediately subsequent to the step S',
19 while setting the node N' as the context node (Steps 2506).
20 Then, the tuple without possibility of such input is removed
21 from the saved set (Step 2507).

22 Next, a description will be made regarding a predicate,
23 which has been omitted in the operation described above.
24 Upon evaluation of the nodes in the above-described

1 operations, evaluation of the predicate is performed if the
2 current sub tree has enough information to evaluate the
3 predicate. Otherwise information, which indicates that the
4 evaluation of the predicate has not been performed, is added
5 to the node and the saved set. Then, the above-described
6 operation is continued based on an assumption that the
7 evaluation has been performed successfully. However, regard-
8 ing a node which depends on a node on which the evaluation of
9 the predicate is not completed, output of such a node will be
10 withheld. Evaluation of an element in the saved set, whose
11 the predicate is not evaluated, will be carried out at a time
12 point when conditions for the evaluation of the predicate are
13 fulfilled by subsequent input. When the location path is
14 used as the predicate, the object of evaluation is changed to
15 evaluation of the predicate, and the processing described in
16 Figure 23 to Figure 25 is applied thereto.

17 When the evaluation of the predicate is performed
18 successfully at last, the node depending on the success in
19 the evaluation is outputted as an evaluation result. On the
20 contrary, when the evaluation is failed, the relevant node is
21 removed from the saved set, and all the operations performed
22 on the assumption of the success in the evaluation of the
23 node are canceled.

24 Next, a description will be made regarding the

1 evaluation of the XPath according to this embodiment based on
2 a concrete example.

3 Figure 26 is a view showing an XML document to be
4 processed, SAX event strings thereof, and a document tree.

5 An assumption is made herein that the following XPath
6 is provided to the XPath evaluating unit 50:

7 /a/b/preceding-sibling::c/following-sibling::d

8 That is, this XPath is evaluated successfully upon detection
9 of a sibling node d subsequent to a sibling node c that is
10 precedent to a node b being a child node of a node a.

11 Figure 27 to Figure 36 show the SAX event strings with
12 respect to the XML document of Figure 26, the document tree
13 corresponding thereto, and aspects of transition of the saved
14 set obtained by the evaluation of the XPath using the
15 document tree in a sequential order.

16 To begin with, a SAX event "startDocument" indicating a
17 start of a document is inputted, and a root node "doc" is
18 constructed as a document tree corresponding thereto (Figure
19 27). A tuple (a set of a step ID and a node ID) <0, doc> is
20 inputted to the saved set. Subsequently, a SAX event
21 "startElement:"a"" indicating a start of an element (a node)
22 is inputted, and a child node "a" is added to the root node

1 "doc" (Figure 28). Meanwhile, since a step 1 of the XPath is
2 established, a tuple <1, "a"> is inputted to the saved set.

3 Next, a SAX event "startElement:"c"" is inputted and a
4 child node "c" is added to the node "a" (Figure 29). Subse-
5 quently, a SAX event "endElement:"c"" indicating an end of
6 the element is inputted (Figure 30). Next, when a SAX event
7 "startElement:"b"" is inputted, a child node "b" is added to
8 the node "a" (Figure 31). In this event, a step 2 of the
9 XPath is established by generation of the node "b" and a step
10 3 of a reverse axis (preceding-sibling) is immediately estab-
11 lished. Therefore, a tuple concerning the step 2 is not
12 retained in the saved set and a tuple <3, "c"> concerning the
13 step 3 is inputted to the saved set instead.

14 Thereafter, a SAX event "endElement:"b"" is inputted
15 (Figure 32). Subsequently, when a SAX event
16 "startElement:"d"" is inputted, a child node "d" is added to
17 the node "a" (Figure 33). In this event, a step 4 (the last
18 step) of the XPath is established by generation of the node
19 "d", and the XPath is thereby evaluated as established.
20 Therefore, the node "d" is outputted from the evaluation
21 executing unit 54 of the XPath evaluating unit 50 to the
22 application executing unit 30 as an evaluation event (see
23 Step 2306 in Figure 23).

24 Thereafter, a SAX event "endElement:"d"" is inputted

1 (Figure 34), and a SAX event "endElement:"a"" is inputted
2 subsequently. Since there is no possibility that a new child
3 node is added to the node "a", the tuple <1, "a"> of the step
4 1 is removed from the saved set (Figure 35; see Steps 2503
5 and 2504 in Figure 25). Furthermore, a SAX event "endDocu-
6 ment" indicating an end of the document is inputted, whereby
7 the input of the event strings of the XML document subject to
8 processing is completed (Figure 36).

9 Figure 37 is a flowchart which explains streaming
10 processing of an XML document according to this embodiment.
11 As shown in Figure 37, when the XML document to be processed
12 is inputted and the XML events are sent from the XML server
13 10 (Step 3701), the XPath evaluating unit 50 firstly
14 constructs a document tree based on the obtained XML events
15 (Step 3702). This document tree is updated every time an XML
16 document is inputted in order and is eventually constructed
17 into the document tree which corresponds to the document
18 structure of the XML document to be processed. Then, the
19 XPath is evaluated by the evaluation executing unit 54 of the
20 XPath executing unit 50 by use of this document tree, and the
21 information concerning partially established portions (steps)
22 are accumulated in the saved set as necessary (Step 3703).
23 Then, the input of the XML events, the construction of the
24 document tree, and the evaluation of the XPath are repeated

1 until the entire XPath is established. When the entire XPath
2 is evaluated as established, the evaluation event is sent to
3 the application executing unit 30 (Steps 3704 and 3705).

4 The application executing unit 30 obtains the XML
5 events outputted from the XML parser 10 and the evaluation
6 event outputted from the evaluation executing unit 54 of the
7 XPath evaluating unit 50. The application executing unit 30
8 serially processes the XML events in accordance with the
9 evaluation event.

10 When all the processing from Steps 3701 to 3705 is
11 executed for all the XML events regarding the XML document to
12 be processed which are sent from the XML parser 10, the
13 streaming processing of the XML document according to this
14 embodiment is completed (Step 3706).

15 The present invention has been described with the
16 foregoing three embodiments. In these embodiments, the
17 partial evaluation of the XPath is serially performed with
18 respect to the XML event strings which are inputted in the
19 streaming format, and the information concerning the partial
20 evaluation is accumulated, whereby the evaluation result of
21 the entire XPath is eventually obtained.

22 In Embodiments 1 and 2, the XPath is converted into the
23 data structure so that partial evaluation can be performed in
24 response to the XML event strings according to the streaming

1 format, and then the evaluation is serially performed in
2 response to the input of the respective XML events. On the
3 other hand, in Embodiment 3, the document tree of the XML
4 document is constructed while inputting the XML event strings
5 in the streaming format, and the XPath is evaluated sequen-
6 tially by use of this document tree.

7 In Embodiments 1 and 2, the evaluation concerning the
8 step of the reverse axis is only possible under a very
9 limited condition because information concerning the XML
10 events which have been evaluated is lost after the
11 evaluation. On the contrary, in Embodiment 3, there is no
12 limitation for the evaluation concerning the step of the
13 reverse axis because the information on the XML event which
14 have been evaluated is preserved as the document tree even
15 after the evaluation.

16 Moreover, in any of the Embodiments 1 to 3, when a
17 judgment is made that the XPath is established before input-
18 ting all the XML event strings, it is possible to start
19 processing by the application executing unit 30 at that point
20 (the streaming processing). However, there is a possibility
21 in Embodiment 3 that a time difference is caused between
22 notification of the XML event and notification of the evalua-
23 tion event if the XPath includes a step of reverse axis.
24 Moreover, the notification of the evaluation event is not

1 always performed in accordance with the document order (the
2 order of initial letters in XML expression of the respective
3 nodes which appear in the XML expressions of the document) in
4 this case. Therefore, attention is required when the
5 processing by the application executing unit 30 is the one
6 that changes the processing of the XML event in accordance
7 with the evaluation event. Embodiments 1 and 2 do not have
8 such limitation.

9 Therefore, Embodiment 1 or 2 is applied when the XPath
10 does not include a step of a reverse axis. Even when the
11 XPath includes a step of a reverse axis, it is also possible
12 to use Embodiment 3 if the processing by the application
13 executing unit 30 is not the one that changes the processing
14 of the XML event in accordance with the evaluation event.
15 Also, if the processing by the application executing unit 30
16 is the one that changes the processing of the XML event in
17 accordance with the evaluation event, as long as a delay in
18 the processing is allowed, the Embodiment 3 can be used.
19 Such a choice of method becomes possible.

20 In Embodiment 2, the inputted XML event strings are
21 directly stored into the stack as the string information and
22 are just compared with the contents of storage in the stack
23 for expressing the XPath. Accordingly, Embodiment 2 requires
24 very small memory usage and short time for processing.

1 Moreover, Embodiment 2 can be implemented relatively easily.
2 Therefore, when any of the embodiments is applicable, it is
3 preferable to use Embodiment 2.

4 It is needless to say that the technical scope of the
5 present invention will not be limited to the above-described
6 three embodiments, and that the present invention includes
7 various modifications which are identical in terms of the
8 technical idea thereof.

9 As described above, according to the present invention,
10 it is possible to realize an analyzing system and an analyz-
11 ing method for evaluating the XPath while subjecting an XML
12 document to streaming processing.

13 Although the preferred embodiments of the present
14 invention have been described in detail, it should be under-
15 stood that various changes, substitutions, and alternations
16 can be made therein without departing from spirit and scope
17 of the inventions as defined by the appended claims. Varia-
18 tions described for the present invention can be realized in
19 any combination desirable for each particular application.
20 Thus particular limitations, and/or embodiment enhancements
21 described herein, which may have particular advantages to a
22 particular application need not be used for all applications.
23 Also, not all limitations need be implemented in methods,
24 systems and/or apparatus including one or more concepts of

1 the present invention.

2 The present invention can be realized in hardware,
3 software, or a combination of hardware and software. A
4 visualization tool according to the present invention can be
5 realized in a centralized fashion in one computer system, or
6 in a distributed fashion where different elements are spread
7 across several interconnected computer systems. Any kind of
8 computer system - or other apparatus adapted for carrying out
9 the methods and/or functions described herein - is suitable.
10 A typical combination of hardware and software could be a
11 general purpose computer system with a computer program that,
12 when being loaded and executed, controls the computer system
13 such that it carries out the methods described herein. The
14 present invention can also be embedded in a computer program
15 product, which comprises all the features enabling the imple-
16 mentation of the methods described herein, and which - when
17 loaded in a computer system - is able to carry out these
18 methods.

19 Computer program means or computer program in the
20 present context include any expression, in any language, code
21 or notation, of a set of instructions intended to cause a
22 system having an information processing capability to perform
23 a particular function either directly or after conversion to
24 another language, code or notation, and/or reproduction in a

1 different material form.

2 Thus the invention includes an article of manufacture
3 which comprises a computer usable medium having computer
4 readable program code means embodied therein for causing a
5 function described above. The computer readable program code
6 means in the article of manufacture comprises computer
7 readable program code means for causing a computer to effect
8 the steps of a method of this invention. Similarly, the
9 present invention may be implemented as a computer program
10 product comprising a computer usable medium having computer
11 readable program code means embodied therein for causing a a
12 function described above. The computer readable program code
13 means in the computer program product comprising computer
14 readable program code means for causing a computer to effect
15 one or more functions of this invention. Furthermore, the
16 present invention may be implemented as a program storage
17 device readable by machine, tangibly embodying a program of
18 instructions executable by the machine to perform method
19 steps for causing one or more functions of this invention.

20 It is noted that the foregoing has outlined some of the
21 more pertinent objects and embodiments of the present inven-
22 tion. This invention may be used for many applications.
23 Thus, although the description is made for particular
24 arrangements and methods, the intent and concept of the

1 invention is suitable and applicable to other arrangements
2 and applications. It will be clear to those skilled in the
3 art that modifications to the disclosed embodiments can be
4 effected without departing from the spirit and scope of the
5 invention. The described embodiments ought to be construed
6 to be merely illustrative of some of the more prominent
7 features and applications of the invention. Other beneficial
8 results can be realized by applying the disclosed invention
9 in a different manner or modifying the invention in ways
10 known to those familiar with the art.